-PART A

1) Write a program for frame sorting technique used in the buffers.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

struct frame
 {
  int seqno;
  char msg[100];
   }m[100];

int main()
{
  int n,i,j,r,s[100],temp;
  char ch[100];
  printf("Enter the number of frames:");
  scanf("%d",&n);
  for(i=0;i<n;i++)
   {
    s[i]=-1;
    m[i].seqno=-1;
   }

  j=0;
 while(j<n)
  {
  r=rand()%n;
  if(s[r]==-1)
   {
    m[j].seqno=r;
    j=j+1;
    s[r]=1;
   }
}

for(i=0;i<n;i++)
 {
   printf("Enter the message:");
   scanf("%s",m[i].msg);
  srand(i);
 }
printf("The arrived frames are:\n");
 for(i=0;i<n;i++)
  {
```

```
   printf("%d\t%s\n",m[i].seqno, m[i].msg);
  }
  for(i=0;i<n;i++)
   {
  for(j=0;j<n-1-i;j++)
   {
   if(m[j].seqno>m[j+1].seqno)
    {
     temp=m[j].seqno;
     m[j].seqno=m[j+1].seqno;
     m[j+1].seqno=temp;
     strcpy(ch,m[j].msg);
     strcpy(m[j].msg,m[j+1].msg);
     strcpy(m[j+1].msg,ch);
    }
  }
  }
printf("The frames in sorted are :\n Sequence number Message \n");
for(i=0;i<n;i++)
  {
  printf("%d\t%s\n",m[i].seqno,m[i].msg);
  }
 printf("\n");
}
```

OUTPUT

```
Enter the number of frames:4
Enter the message:sixth
Enter the message:sem
Enter the message:computer
Enter the message:science
The arrived frames are:
3       sixth
2       sem
1       computer
0       science
The frames in sorted are :
 Sequence number Message
0       science
1       computer
2       sem
3       sixth
```

2. **Write a program for distance vector algorithm to find suitable path for transmission.**

```
#include<stdio.h>
#define INFINITY 999

struct node
{
  int cost;
  int via;
} c[4][4];

int n;


void findpath(int n1,int n2)
{
 int i,t1,t2;
t1=c[n1][n2].via;
if(t1<n2)
   findpath(n1,t1);
if(t1!=n2)
printf("%d-->",t1);

}
void matrix()
{
int i,j,k,cost,x,t;
for(i=0;i<n;i++)
{
 for(j=0;j<n;j++)
  {
    cost=INFINITY;
    if(i!=j)
      {
        for(k=0;k<n;k++)
          {
            if(i!=k)
              {
                x=c[i][k].cost+c[k][j].cost;
                if(cost>x)
                 {
                  cost=x;
                  t=k;
                 }
              }
```

```
          }
        c[i][j].cost=cost;
        c[i][j].via=t;
        }
     else
       {
        c[i][i].cost=0;
        c[i][i].via=i;
       }
   }
}

}
main()
{
int i,j,k,x,t,cost=INFINITY;
int n1,n2,final,t1,t2,next;
printf("Enter the number of nodes:");
scanf("%d",&n);
printf("Enter the edge matrix(enter 999 if nodirect connection)\n");

for(i=0;i<n;i++)
{
  for(j=0;j<n;j++)
   {
    printf("cost[%d][%d]=",i,j);
    scanf("%d",&c[i][j].cost);
    c[i][j].via=INFINITY;
  }
}

printf("starting matrix:(Each entry has cost and via node\n");
for(i=0;i<n;i++)
{
  printf("row %d\t",i);
for(j=0;j<n;j++)
{
  printf("%d,%d\t",c[i][j].cost,c[i][j].via);
}
printf("\n");
}

matrix();

printf("final matrix\n");
for(i=0;i<n;i++)
```

```
{
  printf("row %d\t",i);
 for(j=0;j<n;j++)
  {
  printf("%d,%d\t",c[i][j].cost,c[i][j].via);
  }
 printf("\n");
}

next=1;
while(next)
{
printf("Enter the two node numbers to find the path\n");
printf("Enter the source node:");
scanf("%d",&n1);
printf("Enter the destination node:");
scanf("%d",&n2);
printf("The shortest path to reach %d from %d has cost =%d\n",n2,n1,c[n1][n2].cost);
printf("The path is:\n");
final=c[n1][n2].via;

printf("%d-->",n1);
findpath(n1,n2);
printf("%d",n2);

printf("would you like to continue (0/1)");
scanf("%d",&next);
}
}
```

OUTPUT

```
Enter the number of nodes:4
Enter the edge matrix(enter 999 if nodirect connection)
cost[0][0]=0
cost[0][1]=1
cost[0][2]=999
cost[0][3]=1
cost[1][0]=1
cost[1][1]=0
cost[1][2]=1
cost[1][3]=999
cost[2][0]=999
cost[2][1]=1
cost[2][2]=0
cost[2][3]=999
```

```
cost[3][0]=1
cost[3][1]=999
cost[3][2]=999
cost[3][3]=0
starting matrix:(Each entry has cost and via node
row 0  0,999  1,999  999,999        1,999
row 1  1,999  0,999  1,999  999,999
row 2  999,999        1,999  0,999  999,999
row 3  1,999  999,999        999,999        0,999
final matrix
row 0  0,0      1,1      2,1      1,3
row 1  1,0      0,1      1,2      2,0
row 2  2,1      1,1      0,2      3,0
row 3  1,0      2,0      3,0      0,3
Enter the two node numbers to find the path
Enter the source node:0
Enter the destination node:2
The shortest path to reach 2 from 0 has cost =2
The path is:
0-->1-->2
would you like to continue (0/1) 0
```

3) Write a program for error detecting code using CRC-CCITT (16-bits).

```
#include<stdio.h>
int input[50];
int n;
struct reg
      {
       int bit;
      }r[16];

int xor(int x,int y)
      {
       x+=y;
       if(x==0 || x==2)
         return 0;
       return 1;
      }

void compcrc()
      {
       int lb,x,j,i;
       for(j=0;j<(n+16);j++)
          {
```

```
        lb=r[15].bit;
        for(i=15;i>0;i--)
            {
             r[i].bit=r[i-1].bit;
            }
        r[0].bit=input[j];
        if(lb==1)
          {
           r[12].bit=xor(r[12].bit,lb);
           r[5].bit=xor(r[5].bit,lb);
           r[0].bit=xor(r[0].bit,lb);
          }
        }
      printf("Register content:\n");
      for(i=0;i<16;i++)
        printf("%d",r[i].bit);
      printf("\n");
      for(x=n,j=15;j>=0;x++,j--)
        input[x]=r[j].bit;
      printf("\nThe total message along with crc :\n");
      for(i=0;i<(n+16);i++)
         printf("%d",input[i]);
     }

int  main()
     {
      int i,j,k,x,y;
      for(i=0;i<16;i++)
        r[i].bit=0;
      printf("\nEnter the number of bits in the input:\n");
      scanf("%d",&n);
      printf("\nEnter the bits:\n");
      for(k=0;k<n;k++)
         scanf("%d",&input[k]);
      for(j=n;j<(n+16);j++)
        input[j]=0;
      printf("\nAt sender:\n");
      compcrc();
      for(i=0;i<16;i++)
        r[i].bit=0;
      printf("\nThe data is transmitted\n");
      printf("Do you want to introduce error : 0/1 \n");
      scanf("%d",&x);
      printf("=====**************=====\n");
      if(x==1)
        {
```

```
    for(i=0;i<n+16;i++)
        scanf("%d",&input[i]);
   }
  printf("\nAt receiver:\n");
  compcrc();
  if(x==1)
   {
    printf("\nThere is an error in the data\n");
    printf("\nThe received data : ");
    for(i=0;i<n;i++)
       printf("%d",input[i]);
   }
  else
   {
    printf("\nThere is no error in the data.\n");
    printf("\nThe received data : ");
    for(i=0;i<n;i++)
        printf("%d",input[i]);
   }
  printf("\n");
 }
```

OUTPUT

Enter the number of bits in the input:
18

Enter the bits:
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1

At sender:
Register content:
1111101000100010

The total message along with crc :
11111111111111111101000100010111111
The data is transmitted
Do you want to introduce error : 0/1
0

4) **Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.**

CLIENT SIDE

```
#include<stdio.h>
#include<netdb.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#define BUFLEN 500
int main(int argc,char **argv)
{
 int n,bytes_to_read;
 int sd,port;
 struct hostent *hp;
 struct sockaddr_in server;
 char *host,*bp,rbuf[BUFLEN],sbuf[BUFLEN];
 switch(argc)
 {
   case 2: host=argv[1];
           port=3000;
           break;
   case 3: host=argv[1];
           port=atoi(argv[2]);
             break;
   default:fprintf(stderr,"usage:%s[port]\n",argv[0]);
           exit(1);
 }
 sd=socket(AF_INET,SOCK_STREAM,0);
 printf("sd=%d",sd);
 bzero((char *)&server,sizeof(struct sockaddr_in));
 server.sin_family=AF_INET;
 server.sin_port=htons(port);
```

```
 if((hp=gethostbyname(host))==NULL)
 {
   fprintf(stderr,"can't get server address\n");
   exit(1);
 }
 bcopy(hp->h_addr,(char *)&server.sin_addr,hp->h_length);
 connect(sd,(struct sockaddr *)&server,sizeof(server));
 printf("connected : server addr is: %s\n",hp->h_name);
 printf("Enter the name of files to transmit:");
 gets(sbuf);
 write(sd,sbuf,BUFLEN);
 printf("Recived a file Contents:\n");
 bp=rbuf;
 bytes_to_read=BUFLEN;
 while((n=read(sd,bp,bytes_to_read))>0)
 {
    bp+=n;
    bytes_to_read-=n;
     printf("%s\n",rbuf);
 }
 close(sd);
 return(0);
}
```

SERVER-SIDE

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<fcntl.h>
#define BUFLEN 500

int main(int argc,char** argv)
{
  int n,fd,bytes_to_read;
  int sd,new_sd,client_len,port;
  struct sockaddr_in server,client;
  char *bp,buf[BUFLEN];
  switch(argc)
  {
    case 1:port=3000;
           break;
    case 2:port=atoi(argv[1]);
           break;
    default: fprintf(stderr,"usage:%s[port]\n",argv[0]);
```

```
            exit(1);
  }
  sd=socket(AF_INET,SOCK_STREAM,0);
  bzero((char *)&server,sizeof(struct sockaddr_in));
  server.sin_family=AF_INET;
  server.sin_port=htons(port);
  server.sin_addr.s_addr=htonl(INADDR_ANY);
  bind(sd,(struct sockaddr *)&server,sizeof(server));
  listen(sd,5);
  printf("listening\n");
  while(1)
  {
    client_len=sizeof(client);
    new_sd=accept(sd,(struct sockaddr *)&client,&client_len);
    printf("after accept\n");
    printf("newsd=%d\n",new_sd);
    bp=buf;
    bytes_to_read=BUFLEN;
    while((n=read(new_sd,bp,bytes_to_read))>0)
          {
            bp+=n;
            bytes_to_read-=n;
          }
     fd=open(buf,O_RDONLY);
     if(fd<0)
     {
       strcpy(buf,"requested file does not exists \n");
       write(new_sd,buf,BUFLEN);
       exit(0);
     }
     while((n=read(fd,buf,BUFLEN))>0)
     {
          write(new_sd,buf,BUFLEN);
          printf("the sending data:\n");
     }
    close(new_sd);
  }
  close(sd);
  return(0);
}
```

OUTPUT

SERVER SIDE

listening

```
inside while
after accept newsd = 3
sending  data
hello
inside while
```

CLIENT SIDE

```
sd = 3
enter name of the file to transmit
hello.txt
received contents
hello
```

5) Implement the above program using message queues or FIFOs as IPC channel.

Message Queues

```
#define PERMS 0666
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/msg.h>

struct msg
{
  long mtype;
  char mtext[512];
};

main()
{
  int mid,sfd,x,y,z;
  char c,filename[25];
  struct msg m,buf;
  size_t sz;
  mid=msgget(4231L,0);
  m.mtype=10L;
  strcpy(m.mtext,"text.doc");
  x=msgsnd(mid,&m,sizeof(struct msg),0);
  do
  {
     sz=msgrcv(mid,&buf,sizeof(struct msg),20L,MSG_NOERROR);
     printf("%s",buf.mtext);
  }while(strlen(buf.mtext)!=0);
  z=msgctl(mid,IPC_RMID,(struct msqid_ds *) 0);
  exit(0);
}
```

```
#define PERMS 0666
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/msg.h>

struct msg
{
  long mtype;
  char mtext[512];
};

main()
{
 int mid,sfd,n,y;
 char c[1],filename[25];
 struct msg m,buf;
 size_t x;
 mid=msgget(4231L,PERMS | IPC_CREAT);
 x=msgrcv(mid,&buf,sizeof(struct msg),10L,MSG_NOERROR);
 strcpy(filename,buf.mtext);
 sfd=open(filename,0);
 do
 {
    n=read(sfd,&c,1);
    m.mtype=20L;
    strcpy(m.mtext,c);
    y=msgsnd(mid,&m,1,0);
  }while(n!=0);
 exit(0);
}
```

FIFO
CLIENT SIDE

```
#include<stdio.h>
#include<fcntl.h>
#include<string.h>
#include<sys/types.h>

int main()
{
char buf[200];
int fd1,fd2,n;
printf("Store the value in the file descriper fd1");
fd1=open("WFIFO",O_WRONLY);
```

```
printf("\nEnter the file name");
scanf("%s",buf);
write(fd1,buf,strlen(buf));
close(fd1);
fd2=open("CFIFO",O_RDONLY);
while((n=read(fd2,buf,128))>0)
 write(1,buf,n);
close(fd2);
return 0;
}
```

SERVER-SIDE

```
#include<stdio.h>
#include<fcntl.h>
#include<sys/stat.h>
#include<string.h>
#include<sys/types.h>

int main()
{
char buf[200];
int fd,fd1,fd2,n;
mkfifo("WFIFO",S_IFIFO|0777);
mkfifo("CFIFO",S_IFIFO|0777);
printf("\nServer is started and");
printf("\nwaiting for client request\n");
fd1=open("WFIFO",O_RDONLY);
n=read(fd1,buf,128);
buf[n]='\0';
close(fd1);
fd2=open("CFIFO",O_WRONLY);
if((fd=open(buf,O_RDONLY))<0)
 {
   write(fd2,"File not found",15);
   printf("\nServer terminates");
   exit(0);
 }
while((n=read(fd,buf,128))>0)
 write(fd2,buf,n);
close(fd);
close(fd2);
printf("\nServer terminates");
return 0;
}
```

OUTPUT

SERVER SIDE

server started
server is waiting
.....
server terminated

CLIENT SIDE

enter filename
cs.txt
CLIENT SERVER PROGRAM

6) Write a program for simple RSA algorithm to encrypt and decrypt the data.

```
#include<stdio.h>
typedef unsigned int uint;
uint gcd(uint x,uint y)
{       return y==0? x:gcd(y,x%y);          }

uint multi(uint txt, uint ed, uint n)
{       uint i,rem=1;
        for(i=1; i<=ed; i++)
                rem=(rem*txt)%n;
        return rem;
}

short prime(uint no)
{       uint i;
        for(i=2; i<=no/2; i++)
                if(no%i==0) return 1;
        return 0;
}

int main()
{       char msg[100];
        uint pt[100],ct[100],n,d,e,p,q,z,i,len;

        do{
        printf("\nEnter 2 large prime numbers p & q:\n");
        scanf("%d %d",&p,&q);
        }while(prime(p) || prime(q));

        n=p*q;
```

```
        z=(p-1)*(q-1);

        do
        {       printf("\nEnter prime value of e relative to %d(z):",z);
                scanf("%d",&e);
        }while(gcd(e,z)!=1 || e>n);

        for(d=2;d<z;d++)
                if((e*d)%z == 1)
                        break;

        printf("Enter the Message\n");              //get message from keybrd.
        len=read(1,msg,100)-1;

        for(i=0;i<len;i++)                  //store it in plain text array
                pt[i]=msg[i];

        printf("\n Cipher Text=");
        for(i=0;i<len;i++)                  //convert plain to cipher text
                printf("%d ",ct[i]=multi(pt[i],e,n));

        printf("\n Plain Text=");
        for(i=0;i<len;i++)                  //convert cipher to plain text
                printf("%c",multi(ct[i],d,n));
}
```

OUTPUT

```
Enter 2 large prime numbers p & q:
101 103

Enter prime value of e relative to 10200(z):107
Enter the Message
welcome

 Cipher Text=654 1616 8069 3508 9181 2008 1616
 Plain Text=welcome
```

7) Write a program for Congestion control using the leaky bucket algorithm.

```
#include<stdio.h>
int rand(int a)
{       int rn=(random()%10)%a;
        return rn==0?1:rn;
}
```

```
int main()
{       int packet_sz[5],i,clk,b_size,o_rate,p_sz_rm=0,p_sz,p_time;
        for(i=0;i<5;++i)
                packet_sz[i]=rand(6)*10;
        for(i=0;i<5;++i)
                printf("packet[%d]:%d bytes\t",i,packet_sz[i]);
        printf("\nEnter the Output rate:");
        scanf("%d",&o_rate);
        printf("Enter the Bucket Size:");
        scanf("%d",&b_size);
        for(i=0; i<5; ++i)
        {       if((packet_sz[i]+p_sz_rm) > b_size)
                        if(packet_sz[i] > b_size)
                                printf("\n\nIncomming packet size (%d) is Greater than bucket
capacity-PACKET REJECTED",packet_sz[i]);
                        else
                                printf("\n\nBucket capacity exceeded-REJECTED!!");
                else
                {       p_sz_rm+=packet_sz[i];
                        printf("\n\nIncomming Packet size: %d",packet_sz[i]);
                        printf("\nBytes remaining to Transmit: %d",p_sz_rm);
                        p_time = rand(4)*10;
                        printf("\nTime left for transmission: %d units",p_time);
                        for(clk=10; clk<=p_time; clk+=10)
                        {       sleep(1);
                                if(p_sz_rm)
                                {       if(p_sz_rm <= o_rate)
                                                printf("\n  Packet of size %d
Transmitted",p_sz_rm),
                                                p_sz_rm=0;
                                        else
                                                printf("\n  Packet of size %d Transmitted",o_rate),
                                                p_sz_rm -= o_rate;
                                        printf("----Bytes Remaining after Transmission:
%d",p_sz_rm);
                                }
                                else
                                        printf("\n  No packets to transmit!!");
                                printf("   Time Left:%d",p_time-clk);
}       }       }       }
```

OUTPUT

```
packet[0]:30 bytes
packet[1]:10 bytes
packet[2]:10 bytes
```

```
packet[3]:50 bytes
packet[4]:30 bytes
Enter the Output rate:10
Enter the Bucket Size:25

Incomming packet size (30) is Greater than bucket capacity-PACKET REJECTED

Incomming Packet size: 10
Bytes remaining to Transmit: 10
Time left for transmission: 10 units
  Packet of size 10 Transmitted----Bytes Remaining after Transmission: 0   Time Left:0

Incomming Packet size: 10
Bytes remaining to Transmit: 10
Time left for transmission: 20 units
  Packet of size 10 Transmitted----Bytes Remaining after Transmission: 0   Time Left:10
  No packets to transmit!!   Time Left:0

Incomming packet size (50) is Greater than bucket capacity-PACKET REJECTED

Incomming packet size (30) is Greater than bucket capacity-PACKET REJECTED
```

8) Write a program for Hamming Code generation for error detection and correction.

```
#include<stdio.h>
#include<stdlib.h>
int power(int x, int y)
{
  int i, res=1;
  for(i=1;i<y;i++)
  res=x*res;
  return res;
}

main()
{ int input[15],m,r=0,count=0,i=0,z=0,j=0,n,t[15],k=0,a,b,c,cnt=0,reg=0;
printf("enter the number of bits");
scanf("%d",&m);
printf("enter the %d bits:",m);
for(i=m-1;i>=0;i--)
scanf("%d",&input[i]);
while(!(power(2,r)>=(m+r+1)))
{ r++; }

for(i=1;i<=(m+r);i++)
```

```
{  if(i==power(2,k))
   {   t[i]=0;
       k++;
    }
    else
     t[i]=input[j++];
}
printf("\n the actual message is");
for(i=(m+r);i>0;i--)
printf("%d",t[i]);
n=1;
while(n<=power(2,r))
{  i=n;
   while(i<=m+r)
   {   for(j=0;j<n;j++)
       {  if((i+j)<=(m+r)&& t[i+j]==1)
            count++;
          }
          i=i+2*n;
   }
if(count%2 !=0)
t[n]=1;
n=n*2;
count=0;
}

printf("data transmitted");
for(i=(m+r);i>0;i--)
printf("%d",t[i]);
printf("enter the data transmitted with one bit errror");
for(i=(m+r);i>0;i--)
scanf("%d",&t[i]);
for(i=(m+r);i>0;i--)
printf("%d",t[i]);
printf("\n the errored message is:");
for(i=(m+r);i>0;i--)
printf("%d",t[i]);
n=1;
while(n<=power(2,r))
{  i=n;
   while(i<=m+r)
   {   for(j=0;j<n;j++)
       {  if((i+j)<=(m+r)&& t[i+j]==1)
            cnt++;
          }
          i=i+2*n;
```

```
   }
if(cnt%2 !=0)
{ reg+=n; }
n=n*2;
cnt=0;
}
if(reg==0)
{
printf("no error");
}
else
{printf("error in position %d",reg);
}
}
```

OUTPUT

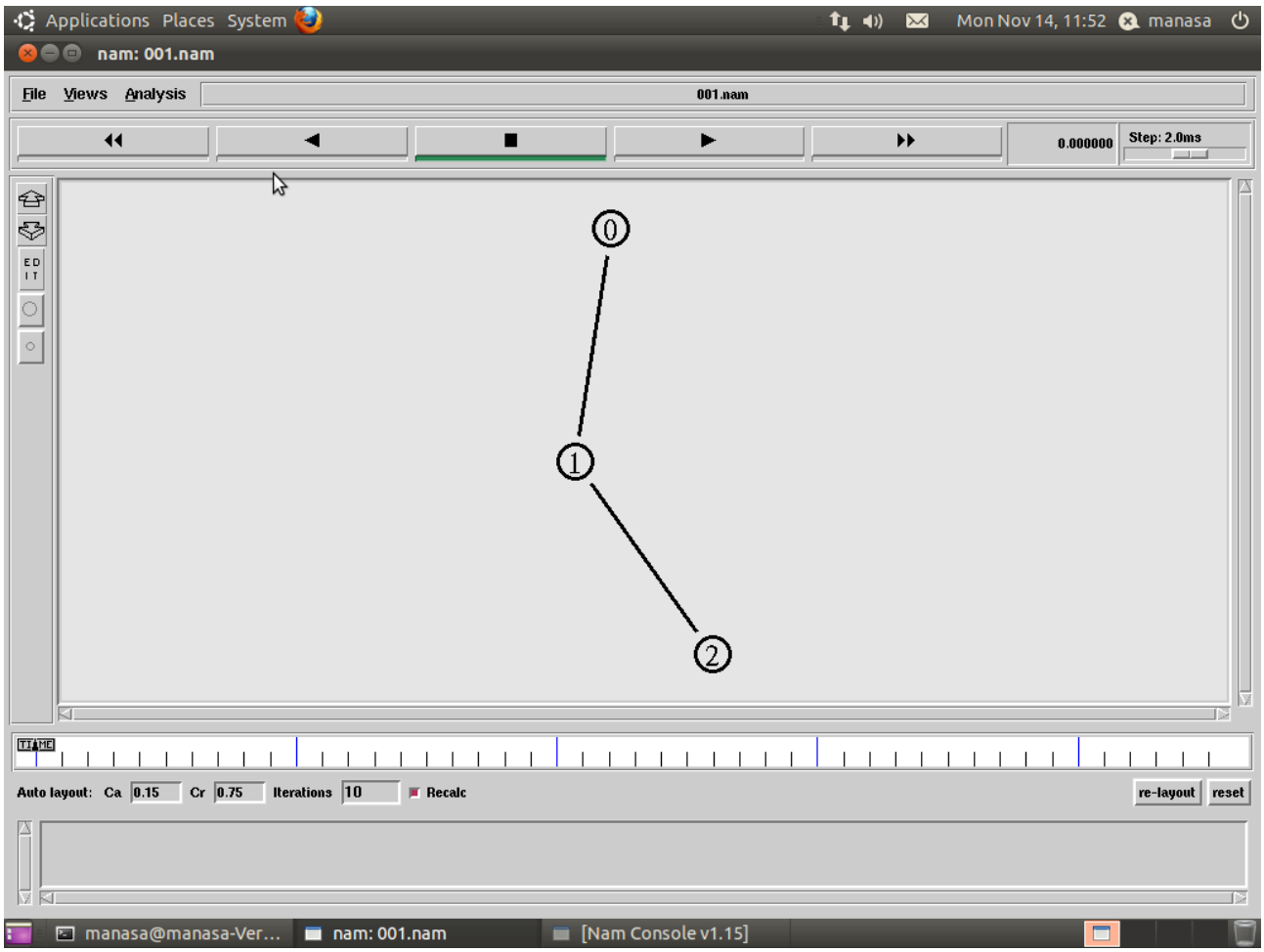
```
enter the number of bits 7
enter the 7 bits: 1 0 0 1 0 0 0
the actual message is 1 0 0 0 1 0 0 0 0 0 0
data transmitted
1 0 0 1 1 0 0 1 0 0 0
enter the data transmitted with one bit errror
1 0 0 1 0 0 0 1 0 0 0
the errored message is:
1 0 0 1 0 0 0 1 0 0 0
error in position 7
```

PART B

1. **Simulate a three nodes point to point network with duplex links between them. Set the queue size and vary the bandwidth and find the number of packets dropped.**

```
set ns [new Simulator]
set nf [open 001.nam w]
$ns namtrace-all $nf
set tf [open 001.tr w]
$ns trace-all $tf
proc finish {} {
        global ns nf tf
        $ns flush-trace
        close $nf
```

```
        close $tf
    exec nam 001.nam &
    exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns queue-limit $n0 $n1 50
$ns queue-limit $n1 $n2 50
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set null0 [new Agent/Null]
$ns attach-agent $n2 $null0
$ns connect $udp0 $null0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"
$ns run
```



2. **Simulate a four node point-to-point network, and connect the links as follows: n0 – n2, n1-n2 and n2-n3. Apply TCP agent between n0-n3 and UDP agents changing the parameter and and find the number of packets dropped.**

```
set ns [new Simulator]
set nf [open 002.nam w]
$ns namtrace-all $nf
set tf [open 002.tr w]
$ns trace-all $tf
proc finish {} {
        global ns nf tf
        $ns flush-trace
        close $nf
```

```
        close $tf
    exec nam 002.nam &
    exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail
$ns queue-limit $n0 $n2 50
$ns queue-limit $n1 $n2 50
$ns queue-limit $n2 $n3 50
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
$ns connect $tcp0 $sink0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
Agent/TCP set packetSize_ 1000
set udp0 [new Agent/UDP]
$ns attach-agent $n1 $udp0
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0
$ns connect $udp0 $null0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
```

$ns at 0.75 "$ftp0 start"

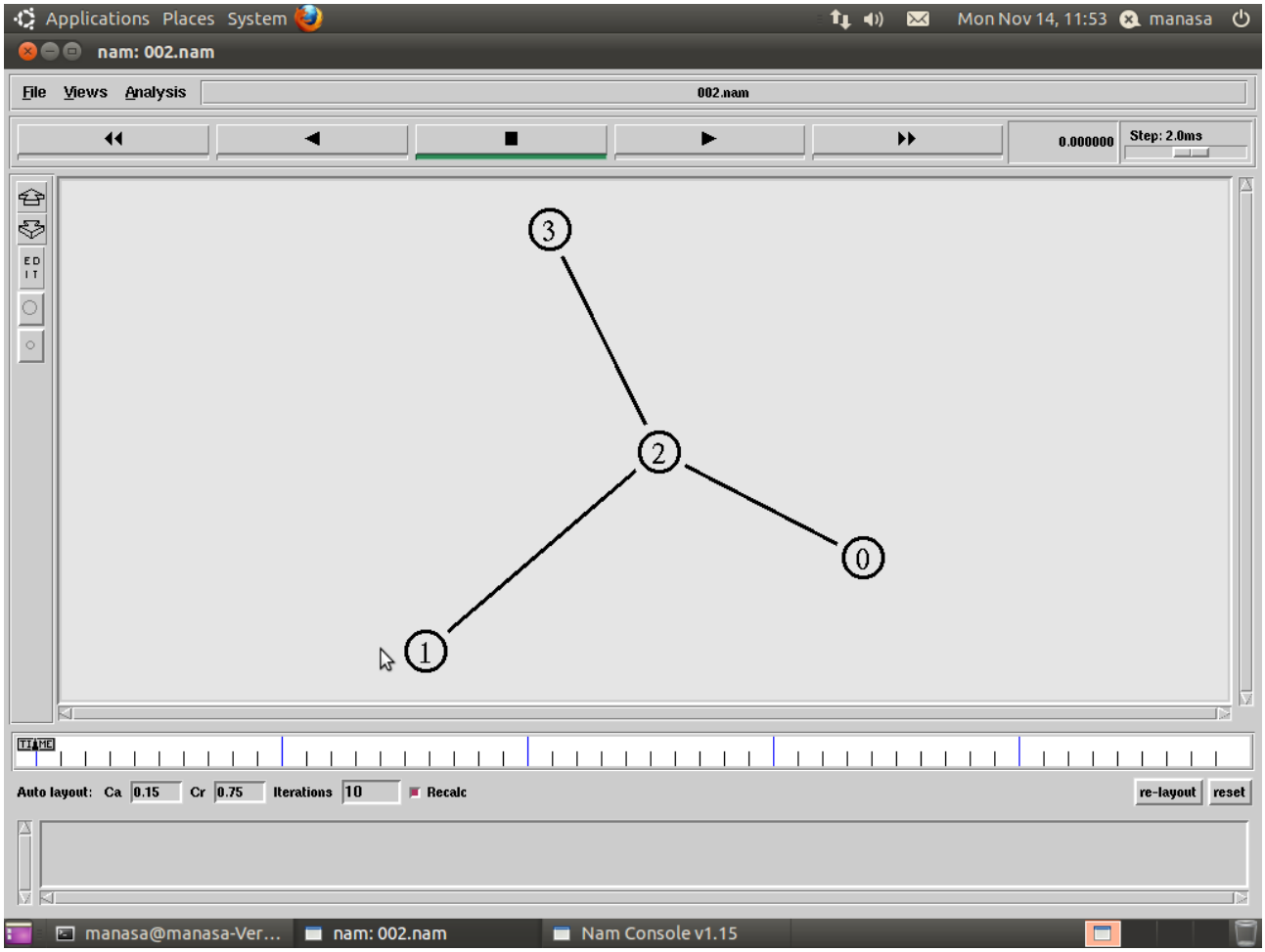
$ns at 4.75 "$ftp0 stop"

$ns at 0.5 "$cbr0 start"

$ns at 4.5 "$cbr0 stop"

$ns at 5.0 "finish"

$ns run



3) Simulate the different types of Internet traffic such as FTP and TELNET over a network and find the number of packets dropped.

set ns [new Simulator]

set nf [open 003.nam w]

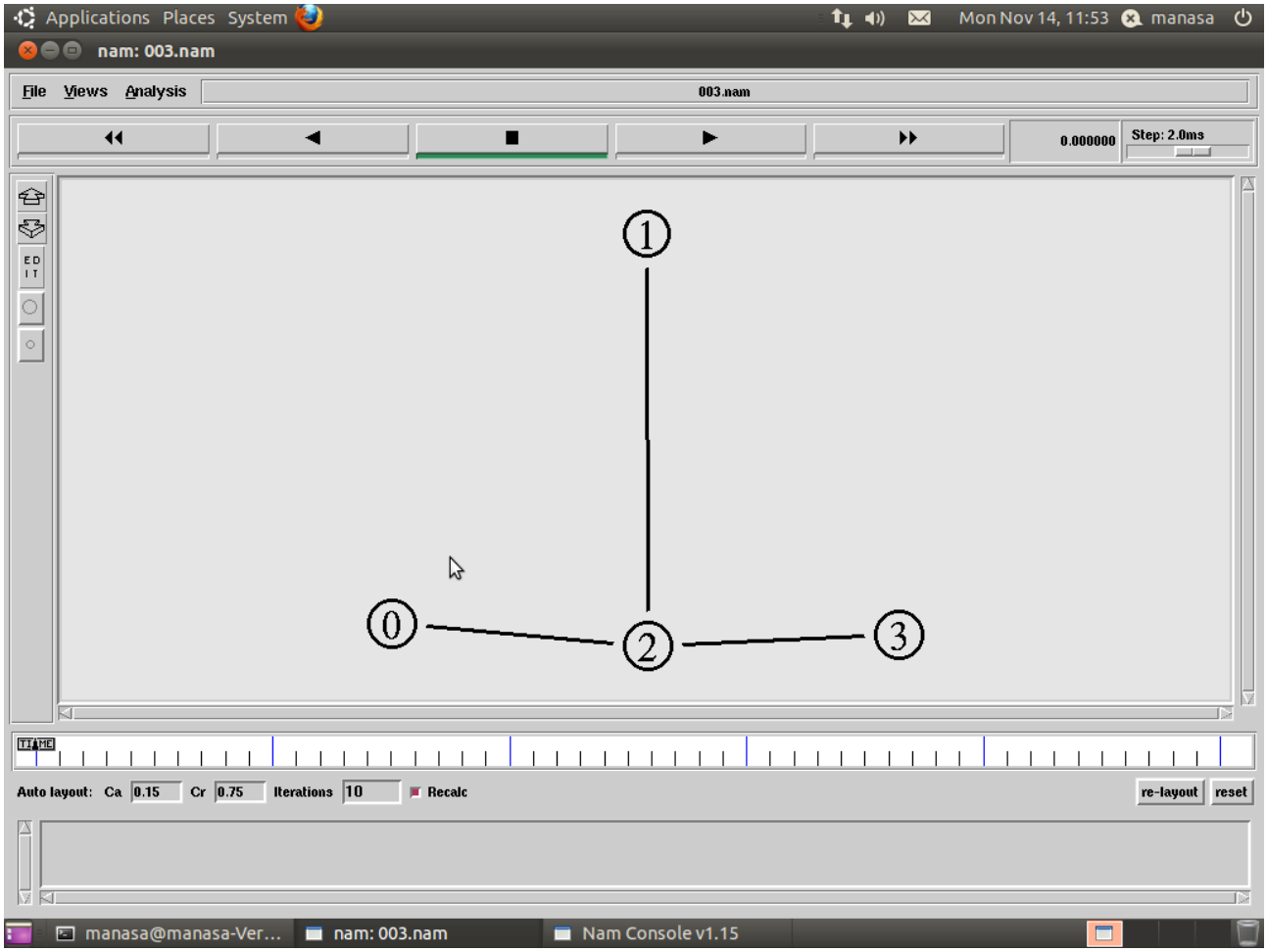
$ns namtrace-all $nf

set tf [open 003.tr w]

$ns trace-all $tf

```
proc finish {} {
        global ns nf tf
        $ns flush-trace
        close $nf
            close $tf
        exec nam 003.nam &
        exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail
$ns queue-limit $n0 $n2 50
$ns queue-limit $n1 $n2 50
$ns queue-limit $n2 $n3 50
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
$ns connect $tcp0 $sink0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
Agent/TCP set packetSize_ 1000
set tcp1 [new Agent/TCP]
$ns attach-agent $n1 $tcp1
set sink1 [new Agent/TCPSink]
$ns attach-agent $n3 $sink1
$ns connect $tcp1 $sink1
```

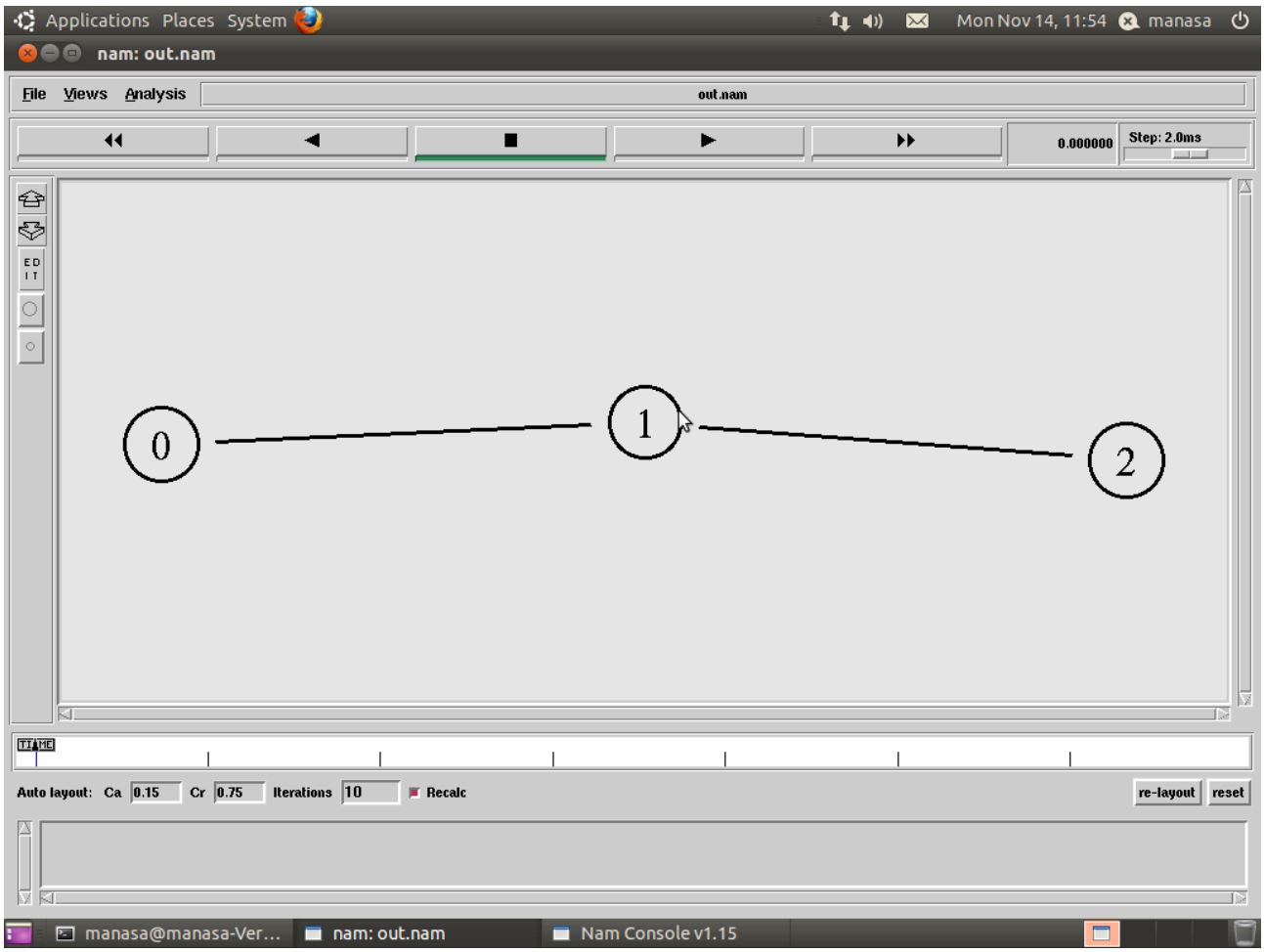
```
set telnet0 [new Application/Telnet]
$telnet0 set interval_ 0.005
$telnet0 attach-agent $tcp1
$ns at 0.75 "$ftp0 start"
$ns at 4.75 "$ftp0 stop"
$ns at 0.5 "$telnet0 start"
$ns at 4.5 "$telnet0 stop"
$ns at 5.0 "finish"
$ns run
```



4) Simulate the transmission of ping messages over a network topology consisting of 3 nodes and find the number of packets dropped due to congestion.

```
set ns [new Simulator]
```

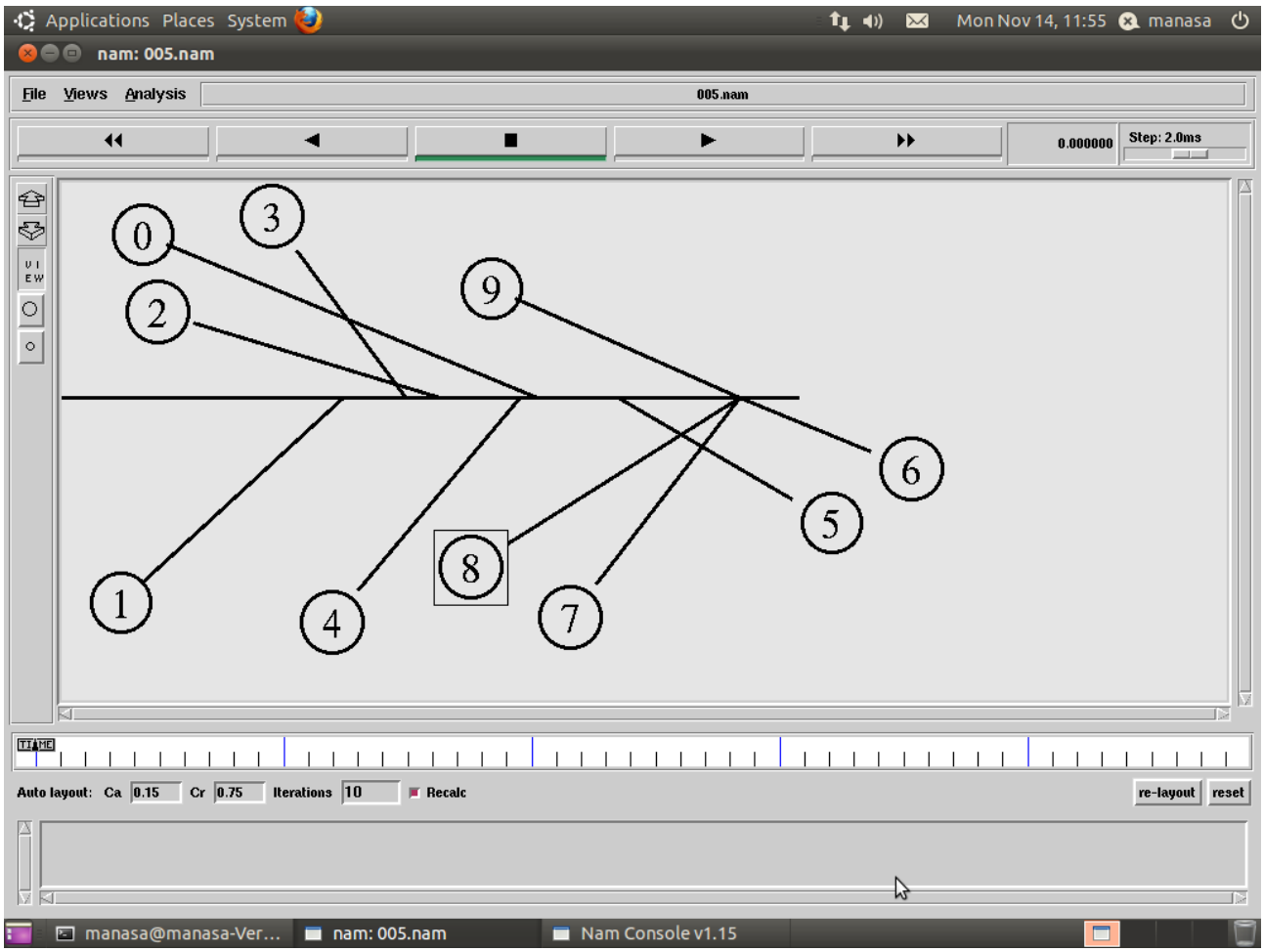
```
set nf [open out.nam w]
$ns namtrace-all $nf
proc finish {} {
        global ns nf
        $ns flush-trace
        close $nf
        exec nam out.nam &
        exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
Agent/Ping instproc recv {from rtt} {
          $self instvar node_
          puts "node [$node_ id] received ping answer from \
            $from with round-trip-time $rtt ms."
}
set p0 [new Agent/Ping]
$ns attach-agent $n0 $p0
set p1 [new Agent/Ping]
$ns attach-agent $n2 $p1
$ns connect $p0 $p1
$ns at 0.2 "$p0 send"
$ns at 0.4 "$p1 send"
$ns at 0.6 "$p0 send"
$ns at 0.6 "$p1 send"
$ns at 1.0 "finish"
$ns run
```



5) **Simulate an Ethernet LAN using N nodes (6-10), change error rate and data rate and find the number of packets dropped for different error rates and data rates.**

```
set ns [new Simulator]
set nf [open 005.nam w]
$ns namtrace-all $nf
set tf [open 005.tr w]
$ns trace-all $tf
proc finish {} {
        global ns nf tf
        $ns flush-trace
        close $nf
```

```
        close $tf
    exec nam 005.nam &
    exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]
set n9 [$ns node]
$ns make-lan "$n0 $n1 $n2 $n3 $n4 $n5 $n6 $n7 $n8 $n9" 100Mb 10ms LL
Queue/DropTail Mac/802_3
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
$ns connect $tcp0 $sink0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
Agent/TCP set packetSize_ 1000
$ns at 0.75 "$ftp0 start"
$ns at 4.75 "$ftp0 stop"
$ns at 5.0 "finish"
$ns run
```



6) Simulate an Ethernet LAN using N nodes and set multiple traffic nodes and find the number of packets dropped.

```
set ns [new Simulator]
set nf [open 006.nam w]
$ns namtrace-all $nf
set tf [open 006.tr w]
$ns trace-all $tf
proc finish {} {
        global ns nf tf
        $ns flush-trace
        close $nf
            close $tf
        exec nam 006.nam &
```

```
    exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]
set n9 [$ns node]
$ns make-lan "$n0 $n1 $n2 $n3 $n4 $n5 $n6 $n7 $n8 $n9" 100Mb 10ms LL
Queue/DropTail Mac/802_3
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
$ns connect $tcp0 $sink0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
Agent/TCP set packetSize_ 1000
set tcp1 [new Agent/TCP]
$ns attach-agent $n1 $tcp1
set sink1 [new Agent/TCPSink]
$ns attach-agent $n3 $sink1
$ns connect $tcp1 $sink1
set telnet0 [new Application/Telnet]
$telnet0 set interval_ 0.005
$telnet0 attach-agent $tcp1
$ns at 0.75 "$ftp0 start"
```

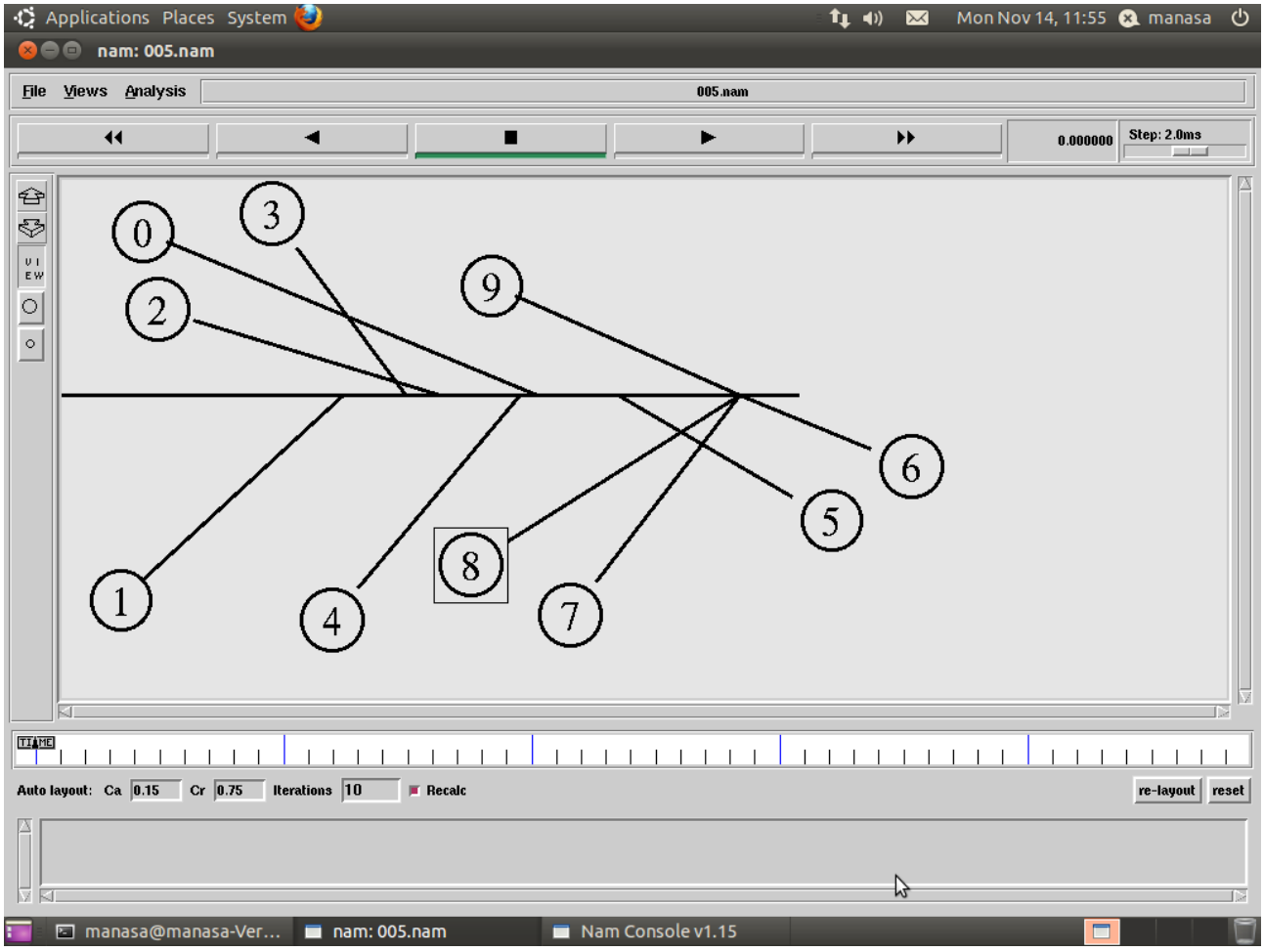
$ns at 4.75 "$ftp0 stop"

$ns at 0.5 "$telnet0 start"

$ns at 4.5 "$telnet0 stop"

$ns at 5.0 "finish"

$ns run



7) Simulate an Ethernet LAN using N nodes and set multiple traffic nodes and plot congestion window for different source/destination.

set ns [new Simulator]

set nf [open 007.nam w]

$ns namtrace-all $nf

set tf [open 007.tr w]

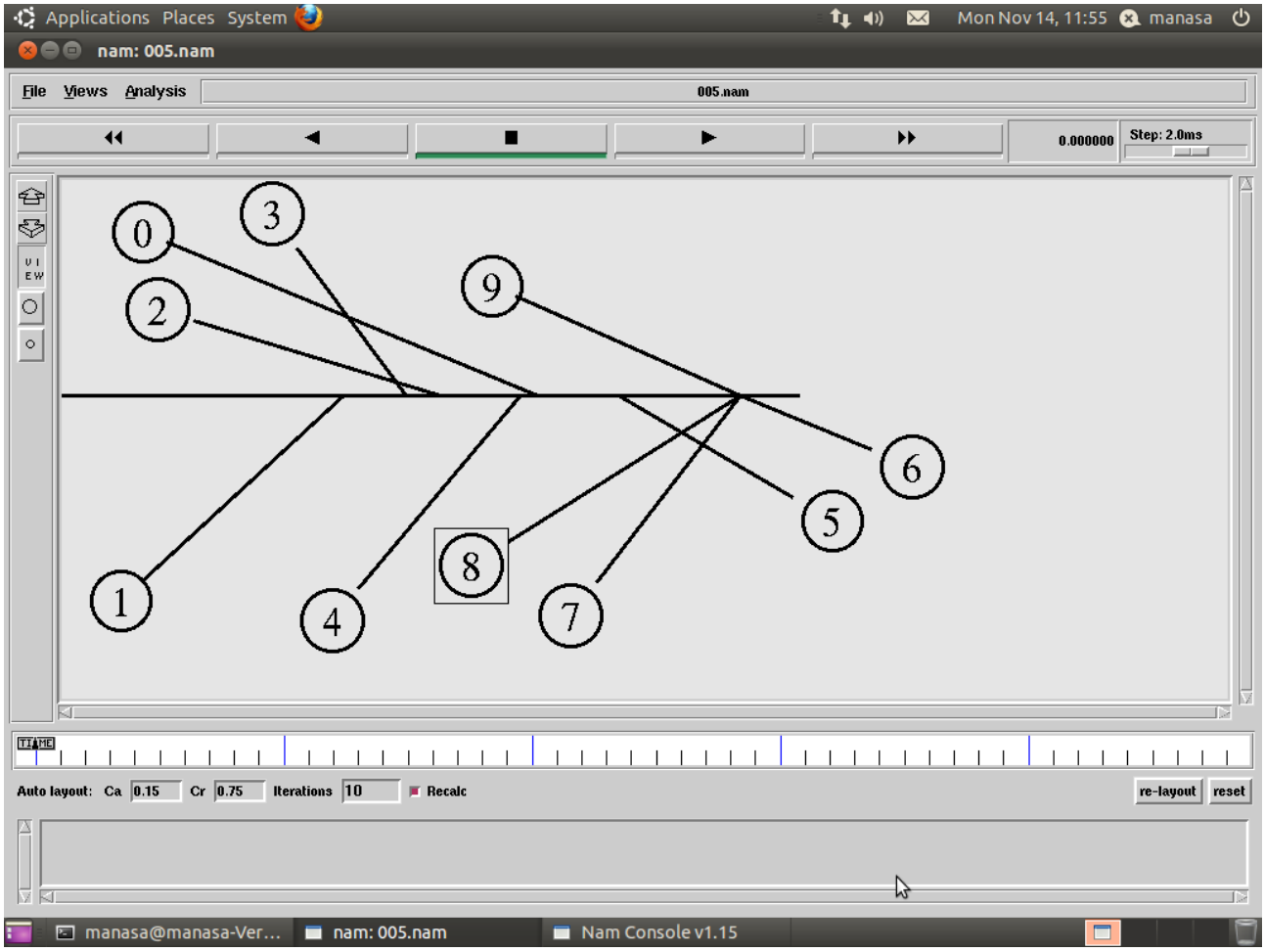
$ns trace-all $tf

proc finish {} {

    global ns nf tf

```
        $ns flush-trace
        close $nf
            close $tf
        exec nam 007.nam &
        exit 0
}
#Create Nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]
set n9 [$ns node]
$ns make-lan "$n0 $n1 $n2 $n3 $n4 $n5 $n6 $n7 $n8 $n9" 100Mb 10ms LL Queue/DropTail
Mac/802_3
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
$ns connect $tcp0 $sink0
#Open a new file to log Congestion Window data
set cfile0 [open tcp0.tr w]
$tcp0 attach $cfile0
$tcp0 trace cwnd_
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
Agent/TCP set packetSize_ 1000
set tcp1 [new Agent/TCP]
$ns attach-agent $n1 $tcp1
```

```
set sink1 [new Agent/TCPSink]
$ns attach-agent $n3 $sink1
$ns connect $tcp1 $sink1
#Open a new file to log Congestion Window data
set cfile1 [open tcp1.tr w]
$tcp1 attach $cfile1
$tcp1 trace cwnd_
set telnet0 [new Application/Telnet]
$telnet0 set interval_ 0.005
$telnet0 attach-agent $tcp1
$ns at 0.75 "$ftp0 start"
$ns at 4.75 "$ftp0 stop"
$ns at 0.5 "$telnet0 start"
$ns at 4.5 "$telnet0 stop"
$ns at 5.0 "finish"
$ns run
```

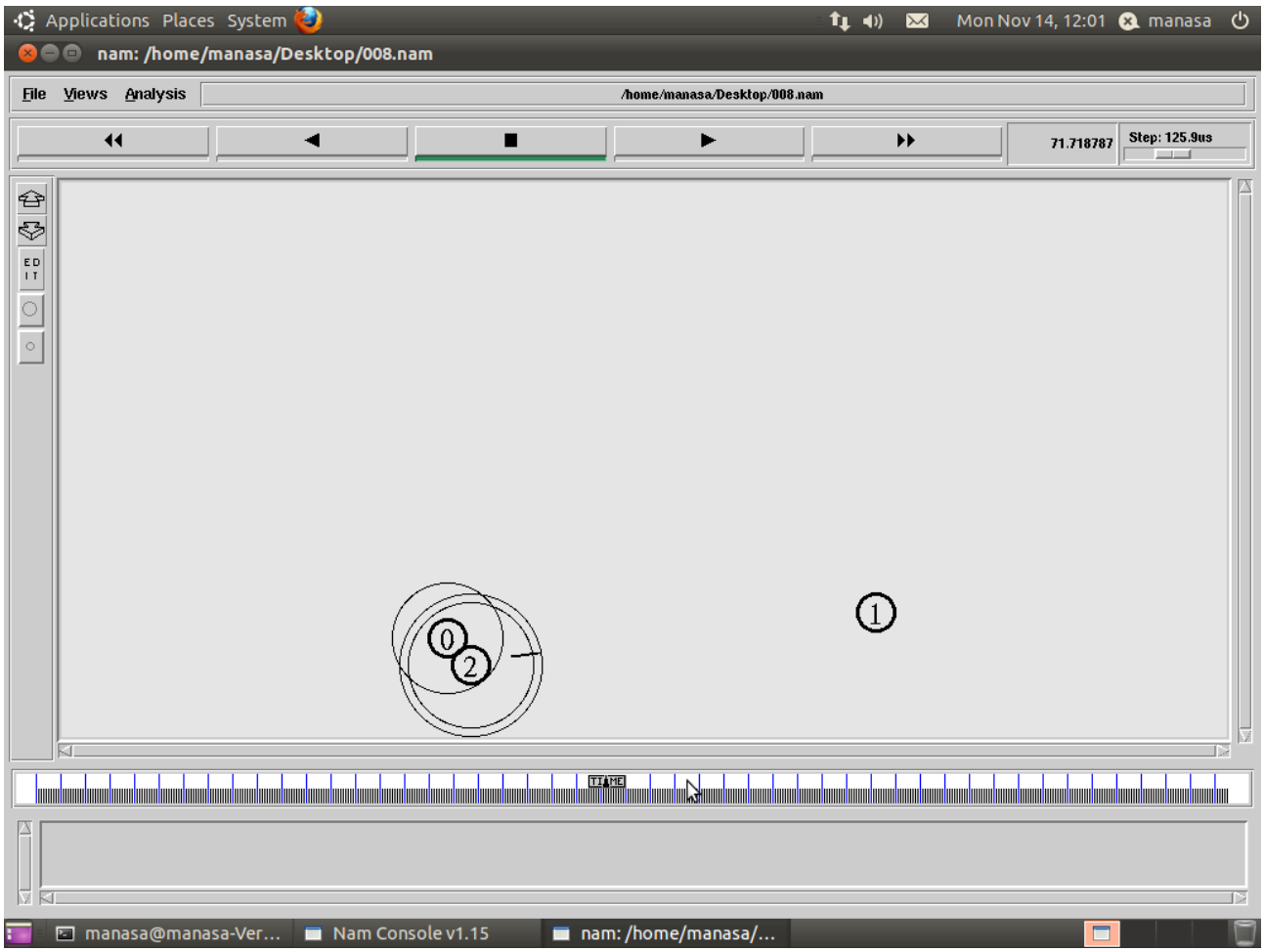


8) Simulate simple wireless scenario consisting of 3 nodes and find the number of packets dropped.

```
set val(chan)          Channel/WirelessChannel    ;#Channel Type
set val(prop)          Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)         Phy/WirelessPhy            ;# network interface type
set val(mac)           Mac/802_11                 ;# MAC type
set val(ifq)           Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)            LL                         ;# link layer type
set val(ant)           Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)        50                         ;# max packet in ifq
set val(nn)            3                          ;# number of mobilenodes
set val(rp)            DSDV                       ;# routing protocol
set val(x)             500
set val(y)             500
```

```
set ns_ [new Simulator]
set tracefd [open 008.tr w]
$ns_ trace-all $tracefd
set namtrace [open 008.nam w]
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)
set chan_1_ [new $val(chan)]
set chan_2_ [new $val(chan)]
$ns_ node-config -adhocRouting $val(rp) \
                 -llType $val(ll) \
                 -macType $val(mac) \
                 -ifqType $val(ifq) \
                 -ifqLen $val(ifqlen) \
                 -antType $val(ant) \
                 -propType $val(prop) \
                 -phyType $val(netif) \
                 -topoInstance $topo \
                 -agentTrace ON \
                 -routerTrace ON \
                 -macTrace ON \
                 -movementTrace OFF \
                 -channel $chan_1_

for {set i 0} {$i < $val(nn) } {incr i} {
            set node_($i) [$ns_ node ]
            $node_($i) random-motion 0       ;# disable random motion
      }

for {set i 0} {$i < $val(nn)} {incr i} {
         $ns_ initial_node_pos $node_($i) 40
}
```

```
$ns_ at 10.0 "$node_(1) setdest 490.0 80.0 20.0"
$ns_ at 10.0 "$node_(0) setdest 50.0 80.0 20.0"
$ns_ at 10.0 "$node_(2) setdest 255.0 100.0 20.0"

set tcp [new Agent/TCP]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp
$ns_ attach-agent $node_(1) $sink
$ns_ connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns_ at 50.0 "$ftp start"

for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at 150.0 "$node_($i) reset";
}
$ns_ at 150.0001 "stop"
$ns_ at 150.0002 "puts \"NS EXITING...\" ; $ns_ halt"
proc stop {} {
    global ns_ tracefd
    close $tracefd
}
puts "Starting Simulation..."
$ns_ run
```



AWK SCRIPT FOR WIRED SCRIPTS(NUMBER OF PACKET DROPS)

```
BEGIN{
#include<stdio.h>
count=0;
}
{
        if($1=="d") #d stands for the packets drops.
        count++
}
END{
printf("The Total no of Packets Dropped due to Congestion : %d\n\n", count)
}
```

AWK SCRIPT FOR WIRELESS SCRIPT(NUMBER OF PACKET DROPS)

```
BEGIN {
count = 0;
}
{
action = $1;
time = $2;
node_id = $3;
layer = $4;
flags = $5;
seqno = $6;
type = $7;
size = $8;
a = $9;
b = $10;
c = $11;
d = $12;
energy = $14;
for(seqno = 0; seqno < 68; seqno++) {
if($1=="D") {

count++;
}
}
}
END {
printf("%d\n",count);
}
```

MARKS DISTRIBUTION

Write Up	10 Marks
C-Program	20 Marks
Simulation Program	10 Marks
Viva	10 Marks
Total	50 Marks